APPENDIX

```
from csv import reader
from math import sqrt
from math import exp
from math import pi
from Tkinter import *

# import main

# Load a CSV file
   def load_csv(filename):
         dataset = list()
         with open(filename, 'r') as file:
               csv_reader = reader(file)
               for row in csv_reader:
                     if not row:
                           continue
                     dataset.append(row)
         return dataset

   # Convert string column to float
   def str_column_to_float(dataset, column):
         for row in dataset:
               row[column] = float(row[column].strip())

   # Convert string column to integer
   def str_column_to_int(dataset, column):
         class_values = [row[column] for row in dataset]
         # print("cv ",class_values)
         unique = set(class_values)
         # print("unik ",unique)
         lookup = dict()

         for i, value in enumerate(unique):
               lookup[value] = i
               # print("lukap ",lookup)
               print('[%s] => %d' % (value, i))
         for row in dataset:
               row[column] = lookup[row[column]]
         # print("ke ",lookup)
         # print("ead  ",dataset)
         return lookup

   # Split the dataset by class values, returns a dictionary
   def separate_by_class(dataset):
         separated = dict()
         for i in range(len(dataset)):
               vector = dataset[i]
               # print("ea ",vector)
               class_value = vector[-1]
               # print("ae ",class_value)
               if (class_value not in separated):
```

```
52.                     separated[class_value] = list()
53.                 separated[class_value].append(vector)
54.         # print("se", separated)
55.         return separated
56.
57.   # Calculate the mean of a list of numbers
58.   def mean(numbers):
59.         a = sum(numbers)/float(len(numbers))
60.         return a
61.
62.   # Calculate the standard deviation of a list of numbers
63.   def stdev(numbers):
64.         avg = mean(numbers)
65.         print("Mean ", mean(numbers))
66.         variance = sum([(x-avg)**2 for x in numbers]) /
  float(len(numbers)-1)
67.         # print("cobaa ",x)
68.         # print("Variance ",variance)
69.         b = sqrt(variance)
70.         print("Standard Deviation ",b)
71.         return b
72.
73.   # Calculate the mean, stdev and count for each column in a
  dataset
74.   def summarize_dataset(dataset):
75.         summaries = [(mean(column), stdev(column),
  len(column)) for column in zip(*dataset)]
76.         del(summaries[-1])
77.         # print("sum ", summaries)
78.         return summaries
79.
80.   # Split dataset by class then calculate statistics for each
  row
81.   def summarize_by_class(dataset):
82.         separated = separate_by_class(dataset)
83.         summaries = dict()
84.         for class_value, rows in separated.items():
85.               summaries[class_value] = summarize_dataset(rows)
86.               d = summaries
87.               # print('summ ',d)
88.         return d
89.
90.   # Calculate the Gaussian probability distribution function
  for x
91.   def calculate_probability(x, mean, stdev):
92.         exponent = exp(-((x-mean)**2 / (2 * stdev**2 )))
93.         c = (1 / (sqrt(2 * pi) * stdev)) * exponent
94.         print("GausProb : ",c)
95.         return c
96.
97.   # Calculate the probabilities of predicting each class for a
  given row
98.   def calculate_class_probabilities(summaries, row):
```

```
99.         total_rows = sum([summaries[label][0][2] for label in
  summaries])
100.        print("totalrows ",total_rows)
101.        probabilities = dict()
102.        for class_value, class_summaries in summaries.items():
103.              probabilities[class_value]                        =
  summaries[class_value][0][2]/float(total_rows)
104.              for i in range(len(class_summaries)):
105.                    mean, stdev, _ = class_summaries[i]
106.                    probabilities[class_value]               *=
  calculate_probability(row[i], mean, stdev)
107.        print("Probabilities : ",probabilities)
108.        return probabilities
109.
110.  # Predict the class for a given row
111.  def predict(summaries, row):
112.        probabilities                                           =
  calculate_class_probabilities(summaries, row)
113.        best_label, best_prob = None, -1
114.        for class_value, probability in probabilities.items():
115.              if  best_label  is  None  or  probability  >
  best_prob:
116.                    best_prob = probability
117.                    best_label = class_value
118.        # print("final",best_label)
119.        # print("final2",best_prob)
120.        return best_label
121.
122.  def final(label):
123.        if label == 0:
124.              label = "Approved"
125.        elif label == 1:
126.              label = "Disapproved"
127.        return label
128.
129.  def run():
130.        filename = file.get()
131.        dataset = load_csv(filename)
132.        # print("dataset ",dataset)
133.        print("filename: ",filename)
134.        for i in range(len(dataset[0])-1):
135.              str_column_to_float(dataset, i)
136.        # convert class column to integers
137.        str_column_to_int(dataset, len(dataset[0])-1)
138.        # fit model
139.        model = summarize_by_class(dataset)
140.        # print("Summary : ", model)
141.        # define a new record
142.        # row = [2,15000000,2500000,12,2.5,3]
143.
144.        a = 0
145.        b = 0
146.        if str(gender.get()) == "L":
147.              a = 1
```

```
148.        elif str(gender.get()) == "P":
149.              a = 2
150.
151.        if str(guarantee.get()) == "K":
152.              b = 3
153.        elif str(guarantee.get()) == "NK":
154.              b = 4
155.
156.        row                                                    =
  [a,int(submission.get()),int(salary.get()),int(period.get()),fl
  oat(interest.get()),b]
157.        # predict the label
158.        label = predict(model, row)
159.        print("label ",label)
160.        coba = calculate_class_probabilities(model, row)
161.        print("asd ",coba)
162.        key, val = coba.items()[0]
163.        key2, val2 = coba.items()[1]
164.  # printing result
165.        print("The 1st key of dictionary is : " + str(key))
166.        print("The 1st value of dictionary is : " + str(val))
167.        print("The 2st key of dictionary is : " + str(key2))
168.        print("The 2st value of dictionary is : " + str(val2))
169.        print('Data Testing=%s, Predicted: %s, Label: %s' %
  (row, final(label), label))
170.
171.        probA.delete(0, END) # delete whatever's there already
172.        probA.insert(0, str(val))
173.        probA.update()
174.
175.        probD.delete(0, END) # delete whatever's there already
176.        probD.insert(0, str(val2))
177.        probD.update()
178.
179.        pred.delete(0, END) # delete whatever's there already
180.        pred.insert(0, str(final(label)))
181.        pred.update()  #  make  sure  update  is  flushed  /
  immediately visible
182.
183.  window = Tk()
184.
185.  window.title("Welcome")
186.
187.  window.geometry('525x400')
188.
189.  judul = Label(window, text="Klasifikasi Kelayakan Pemberian
  Pinjaman")
190.
191.  judul.grid(column=2, row=0)
192.
193.  judul2 = Label(window, text="Input data training")
194.
195.  judul2.grid(column=2, row=1)
196.
```

```
197.  filetxt = Label(window, text="FileName : ")
198.
199.  filetxt.grid(column=1, row=2)
200.
201.  file = Entry(window,width=20)
202.
203.  file.grid(column=2, row=2)
204.
205.  judul3 = Label(window, text="Input data testing")
206.
207.  judul3.grid(column=2, row=3)
208.
209.  gendertxt = Label(window, text="Jenis Kelamin : ")
210.
211.  gendertxt.grid(column=1, row=4)
212.
213.  gender = Entry(window,width=20)
214.
215.  gender.grid(column=2, row=4)
216.
217.  genderinfo = Label(window, text="L : Laki-laki , P :
  Perempuan")
218.
219.  genderinfo.grid(column=3, row=4)
220.
221.  submissiontxt = Label(window, text="Pengajuan : ")
222.
223.  submissiontxt.grid(column=1, row=5)
224.
225.  submission = Entry(window,width=20)
226.
227.  submission.grid(column=2, row=5)
228.
229.  salarytxt = Label(window, text="Gaji : ")
230.
231.  salarytxt.grid(column=1, row=6)
232.
233.  salary = Entry(window,width=20)
234.
235.  salary.grid(column=2, row=6)
236.
237.  periodtxt = Label(window, text="Jangka : ")
238.
239.  periodtxt.grid(column=1, row=7)
240.
241.  period = Entry(window,width=20)
242.
243.  period.grid(column=2, row=7)
244.
245.  interesttxt = Label(window, text="Bunga : ")
246.
247.  interesttxt.grid(column=1, row=8)
248.
249.  interest = Entry(window,width=20)
```

```
250.
251.  interest.grid(column=2, row=8)
252.
253.  guaranteetxt = Label(window, text="Jaminan : ")
254.
255.  guaranteetxt.grid(column=1, row=9)
256.
257.  guarantee = Entry(window,width=20)
258.
259.  guarantee.grid(column=2, row=9)
260.
261.  guaranteeinfo = Label(window, text="K : Kredit , NK : Non-
  Kredit")
262.
263.  guaranteeinfo.grid(column=3, row=9)
264.
265.  btn = Button(window, text="Run", command=run)
266.  btn.pack()
267.  btn.grid(column=2, row=10)
268.
269.  judul4 = Label(window, text="Result")
270.
271.  judul4.grid(column=2, row=11)
272.
273.  probAtxt = Label(window, text="Probabilitas Approved: ")
274.
275.  probAtxt.grid(column=1, row=12)
276.
277.  probA = Entry(window,width=25)
278.  probA.pack()
279.  probA.grid(column=2, row=12)
280.
281.  probDtxt = Label(window, text="Probabilitas Disapproved: ")
282.
283.  probDtxt.grid(column=1, row=13)
284.
285.  probD = Entry(window,width=25)
286.  probD.pack()
287.  probD.grid(column=2, row=13)
288.
289.  predtxt = Label(window, text="Prediksi : ")
290.
291.  predtxt.grid(column=1, row=14)
292.
293.  pred = Entry(window,width=25)
294.  pred.pack()
295.  pred.grid(column=2, row=14)
296.
297.  window.mainloop()
```

PLAGIARISM CHECK.ORG

0.63% PLAGIARISM APPROXIMATELY

Report #12303413

1. Introduction 1.1Background In Savings and Loans Cooperatives, of course, there will be many incoming loan requests from a variety of different customers. Cooperatives cannot simply provide loans to customers without knowing the background of the customer. The cooperative must first find out what the background, type of work, and some other data are used to make a decision whether the customer is eligible or not to be given a loan. In some situations, the cooperative faces difficulties in determining whether or not a customer is eligible for a loan. To solve this problem, a classification program was created to help determine whether the new customer who applied for the loan was eligible or not to be given a loan. The algorithm used in this research is the Naive Bayes Algorithm. 1.2Problem Formulation The problems raised in this study are: 1. How to implement Naive Bayes algorithm? 2. How to test the accuracy of the application calculations? 3. What is the final result of the data testing? 1.3Scope The limitations of the problems in this study are: 1. This study only uses data provided by Sejahtera Savings and Loans Cooperative from 2016 to 2019. 2. This study used 694 data with data sharing of 641 Approval Data and 53 Disapproval Data. 1.4Objective The purpose of this study is to assist Sejahtera Savings and Loans Cooperative in making decisions whether a new